Norwegian School of Economics

Bergen, Fall 2019

Machine Learning in Default Prediction

The Incremental Power of Machine Learning Techniques in Mortgage Default Prediction

Arvin Matre

Supervisor: Jonas Andersson

Master thesis in Economics and Business Administration Major in Finance

NORWEGIAN SCHOOL OF ECONOMICS

Acknowledgements

This master thesis was written as part of the master's degree in economics and business administration with major in finance at the Norwegian School of Economics.

Working with a large and high dimensional data set this semester has been a rewarding experience. Through this thesis I have developed deep insight into the process of modelling probability of default, both how it is done today and how it may be done in the future.

The employees at a financial services firm that shall remain nameless throughout this thesis were kind enough to let me handle this sensitive dataset. I could not have written this without them. Thank you very much for your generous help.

I would also like to express my gratitude and appreciation to Jonas Andersson, my supervisor for this thesis, for valuable guidance. You have been helpful in answering my questions throughout this semester.

Abstract

In this thesis, alternative machine learning techniques have been used to test if these perform better than a Logistic Regression in predicting default on retail mortgages. It is found that the ROC AUC statistic is slightly better for the advanced machine learning techniques, i.e. the Neural Networks, Support Vector Machines and Random Forests. Importantly, all classifiers are trained on the same variables, which are all Weight of Evidence transformed. This enables us to compare the results and view the incremental predictive power as solely a result of the classifiers. Also, it enables us to use the same methodology for probability of default modelling as practitioners currently use, i.e. with Weight of Evidence transformed variables.

The analysis is based on a dataset with observations on each loan issued from a financial services firm in the market for retail mortgages in the years 2009-2017. After univariate and multivariate analysis, the number of candidate variables are reduced from 549 to 19.

The best model is the deep Neural Network, with an impressive ROC AUC of 0,902. This is very high for prediction of default. Still, the Logistic Regression model also has a very high statistic of 0,882. A more primitive machine learning technique is also included in the analysis, the Decision Tree. As expected, this classifier has the lowest ROC AUC of 0,732.

Through the exploratory analysis with WoE variables interesting relationships are found, which may enjoy some readers.

Keywords – Probability of Default, PD, Mortgage default, Bankruptcy prediction, Weight of Evidence, Basel, IRB, Neural Network, Support Vector Machine, Random Forest, K-Nearest Neighbor, Decision Tree, Logistic Regression, ROC, Confusion Matrix

Contents

List of Figures

List of Tables

1. Introduction and Main Findings

This master thesis aims at providing insights into the potential for alternative machine learning techniques for the estimation of default risk on individual customers in mortgage lending. Logistic regression has been the most common estimation technique for decades, as it has provided a fine balance between predictive ability and ease of interpretation. However, corporations today have more information on customers and their behavior than ever before, opening the question as to if any other machine learning techniques should be used.

Modelling probability of default, or more specifically credit scoring, is the use of statistical inference to transform relevant data into measures that may be used to guide credit decisions (Anderson, 2007). In a sense, it is the further development of more subjective credit scoring techniques, which have been used throughout the centuries, to take advantage of the large amounts of data now available, enabled by the computing power of todays' machines. Credit scoring has been especially helpful in high volume mortgage lending and for smaller businesses, where the cost of making a bank representative do an evaluation of the potential customer has been greater than the potential income – which has led many bank not to issue credit for these customers at all.

There are different types of credit scores, and the most informative separation might be between application scores and behavior scores. Application scores are used for origination of new loans, where data about the borrower's income, size of the loan, previous behavior with other products in the bank etc. are used. Behavior scores are used to guide decision making regarding over-limit management, evaluating the risk of the portfolio and more. This thesis is concerned with developing probability of default (PD) models on a dataset with customers that have been granted a loan, i.e. for applicants.

The predictive models used for credit scoring may be separated into parametric and non-parametric models, where the former models make assumptions about the data, while the second does not (Anderson, 2007). The typical models used by banks to estimate default probabilities are logistic regression models, which are parametric.

In response to the assumptions needed to calculate parametric models, and the need to build models with better predictive abilities, non-parametric models are increasingly considered and

used by financial institutions. Machine learning models are associated with this category of models. There are mainly two drawbacks with these models:

1. Lack of transparency
2. Tendency to overfit

Regulators have strict requirements concerning the interpretability of credit scoring models. Indeed, the burden is on the bank to provide evidence of model interpretability (Finanstilsynet , 2019). This is, without doubt, a strength of the logistic regression models – they are easy to interpret. However, there are some machine learning models that also provide high interpretability, such as decision threes. Concerning the tendency to overfit, this is certainly a potential danger if only a training set was used to create the model. However, all modern statistical and machine learning tools allow for the data to be split into training, validation and testing splits, which should reduce the tendency to overfit. Further, the ability of machine learning models to capture non-linearities and interactions in the data might outweigh these issues.

The dataset used is provided by a financial services firm and contains 549 variables. The dataset is of very high quality and has been used for internal development of models prior to this thesis. It comes pre-cleaned, but some modifications are done. Each entry represents a loan agreement from the financial institution to the private customer. All loans are secured against the property bought, thus we are dealing with mortgages for the retail market.

The variables used to train the models are first chosen based on univariate analysis using the Information Value (IV) statistic, before a multivariate analysis with stepwise regression is performed. Also, new variables are created based on interactions. Following the multivariate analysis, we are left with 19 variables. To create a level playing field, the analysis begins with the same 19 variables used for all estimation techniques. Weight of Evidence transformations are used in the univariate analysis, but also to account for missing values, outliers and ease for interpretation. Then, by comparing the Logistic Regression to other machine learning techniques on the Weight of Evidence transformed variables, the exact increase in predictive power can be inferred. Had the machine learning classifiers been trained on variables that were not transformed the results would not be comparable to the current modelling methodology for probability of default, since there Weight of Evidence transformations are used.

Different statistical and machine learning techniques are used to estimate models for predicting default; Logistic Regression, Decision Trees, Random Forest, Neural Networks, Gradient Boosting, K-Nearest Neighbors and Support Vector Machines. The models are evaluated on their predictive performance using the ROC AUC measure. Although the use of Weight of Evidence transformed variables and logistic regression has been the industry standard, partly because it makes it possible to construct an easy to understand scorecard, it is found that the more advanced machine learning models perform slightly better.

The Neural Network, Support Vector Machine and Random Forest classifiers all perform better than the logistic regression, but only slightly. The deep Neural Network performs best with a ROC AUC of 0.903, compared to 0.883 for the logistic regression. This is in line with the results from previous literature on the subject. For example, West (2000) and Lee et al. (2002) find that Neural Networks perform better than K-Nearest Neighbors, Decision Trees and Logistic Regressions. The Decision Tree has the lowest ROC AUC, which is not suprising given that it is a very simple algorithm.

The thesis is concluded by a statement about the impact these results may have for financial institutions using the estimation techniques. Since the differences between the advanced machine learning techniques and the Logistic Regression are small, it is not obvious that the former should be adopted. There are clearly costs associated with the black-box nature of these advanced machine learning models, and legal risks associated with customers' right to explanation and the general legislation around interpretability of credit scoring algorithms.

Lastly, it should be mentioned that although sentences such as "The Logistic Regression versus the machine learning models" will appear, the Logistic Regression may also be seen as a machine learning model. Indeed, all models considered are classifiers. Thus, the distinction is made purely for pragmatic reasons.

2. Literature Review

2.1 Predictive Performance of Machine Learning Techniques

Thomas (2000) and Crook et al. (2007) gives an account of different publications investigating the use of machine learning techniques for PD modelling. For example, in Baesens (2003) it is observed that while Neural Networks (NN) have not been used much in this setting, it is a very common technique in other areas of banks, particularly fraud detection. When NNs are compared with Logistic Regression, Decision Trees, K-Nearest Neighbours (K-NN) and Support Vector Machines (SVM), it is found that NNs and SVM often perform best. This is also what is found by other authors. For example, comparing NNs with a Logistic Regression, both West (2000) and Lee et al. (2002) find the NN to be superior. Still, K-NN is also found to have high predictive power by Ong et al. (2005).

Table 2.1 displays the error rates for different classifiers in relevant papers. One should be cautious in comparing the results between different papers, since the error rates are defined in slightly different ways, but within paper comparisons are legitimate (bold numbers indicate "best in the given paper"). As can be seen from the table, Logistics Regressions often perform quite well. Thus, one should not exaggerate the impact of an implementation of more advanced machine learning techniques based on the current literature. Also, one should note that if the activation function in the NN is logistic, it very much behaves as a Logistic Regression between the hidden layers. In that way, it might be seen as a generalization of the Logistic Regression.

In the comparison between Logistic Regression and NNs by Desai et al. (1997) the logistic regression is actually found to perform better. However, as is noted by the authors, "the accuracy of neural networks … depends on the values of certain parameters which are under the control of the investigator" (Desai et al., 1997). This might explain why there is a large variation in results from comparing advanced machine learning models and the Logistic Regression.

Decision Trees are rather often found to have low accuracy in prediction (Krauss, 2014). Therefore, a more general type of tree technique is more common in predictions – Random Forests. Random Forest techniques belong to the class of ensemble methods. These methods combine a set of trees, which to some extent overcomes the instability of single trees.

In the general literature there has been many comparisons between classifiers in their predictive power, but relatively few in the area of consumer credit data, and particularly in application scoring. It might be that the data and the abstract data structures in datasets for consumer credit scoring are so different from each other (i.e. from bank to bank) that it is not clear which method is best to use in general (Crook et al., 2007). Indeed, since banks could save a tremendous amount on choosing the right customers to lend to, the best approach might be to try different methods.

Hand (2006) argues that the relative differences in predictive power between the classifiers could be exaggerated. This could for example be a result of the "reject inference" problem. One classifier that performs well on the dataset does not necessarily perform better than other classifiers on through-the-door applicants. Further, Hand argues that the aim of the classifier should be to maximize profit. If an evaluation method such as a profit matrix is used, it might give different results than when using the ROC AUC statistic, which is more common in research.

Table 2.1: Error rates for machine learning techniques in the literature

	Boyle et al. (1992)	Henley (1995)	Desai et al (1997)	Yobas et al. (2000)	West (2000)	Lee et al. (2002)	Baesens (2003)	Ong et al. (2005)
Logistic regression		43,3	67,3		81,8	73,5	79,3	
Decision trees	75,0	43,8		62,3	77,0		77,0	78,4
Neural networks			66,4	62,0	82,6	77,0	79,4	81,7
K-NN					76,7		78,2	82,8
SVM							79,7	

Some authors have tried to combine different classifiers. For example, Kao and Chiu (2001) used a combination of NNs and Decision Trees. Other authors have used Decision Trees to select variables to use in a Logistic Regression.

2.2 The Basel Accords

2.2.1 The Context

The Basel Accords (Basel I, II and III) shall not be a main discussion topic in this thesis. However, it is important to see the context this thesis is written in. In Basel II, the three pillars of sound regulation are described, the first of which is minimum capital requirements (Basel, 2005). The banks may choose for themselves which method to follow to calculate minimum

capital requirements, so called regulatory capital. In the standardized approach, a fixed percentage of outstanding loans is set aside. This percentage varies for different asset classes. This may be seen as the easiest and most primitive approach, but it may be very expensive for the banks in that they hold much capital when it is not needed, and the opposite. In the Internal Ratings Based (IRB) approach, the bank chooses the percentage of total exposure in each asset class to set aside. Expected and unexpected losses are to be calculated, where the second is of much greater importance. Indeed, regulatory capital is only concerned with the unexpected losses. As part of the methodology to calculate unexpected losses, PD models are built. The implication is that when reading this thesis, one should note that the models described are not used by the firm to score customers prior to issuing a loan. The models are used by the firm to set aside enough regulatory capital. Still, the potential for using the models for screening shall be discussed.

2.2.2 Definiton of Default

One should reach a balance between "strict" definitions of default and the concern for having enough observations in the sample with event = 1 (Siddiqi, 2006). Strict definitions would for example allow for more "days delinquent", as then one would be surer that the customer is in default. But doing so would perhaps leave too few event observations in the sample. With PD modeling for banks, the most important consideration to make when defining defaults is the regulatory requirements. Under Basel II, a default event has occurred when either or both of the following conditions are fulfilled:

- "The bank considers that the obligor is unlikely to pay its credit obligations to the banking group in full, without recourse by the bank to actions such as realizing security (if held)"
- "The obligor is past due more than 90 days on any material credit obligation to the banking group. Overdrafts will be considered as being past due once the customer has breached an advised limit or been advised of a limit smaller than current outstanding" (BIS, 2006)

3. Methodology

This chapter is split in three. The first part describes the machine learning techniques that are most common in the literature on credit scoring. In the second part, Weight of Evidence is introduced as a method to transform variables, a common method for capturing non-linear effects in PD modelling. The third part describes the two methods used for evaluation of the models – the confusion matrix and ROC AUC.

3.1 Classification Techniques for Credit Scoring

Table 3.1 gives an account of different predictive models typically used in the literature on machine learning algorithms for PD modelling, and a short description of them. All models except the first are non-parametric. In the sub chapters below, each classification technique will be explained, and the unique hyperparameters to be tuned in the classification techniques shall be emphasized.

Table 3.1: Classification Techniques for Credit Scoring

Predictive model	**Description**
Logistic regression	Regression with probabilistic dependent variable
Decision trees	Sequence of branching operations partitioning the data
Neural Networks	Network of nodes weighing and transforming input
Support Vector Machines	Fitting hyperplanes in the feature space to best classify the dependent variable
Random Forest	Ensemble learning by constructing multiple decision trees
Gradient Boosting	Constructing a model based on ensemble of weaker models with boosting
K-Nearest Neighbors	Classification using the k nearest neighbors in the feature space

3.1.1 Logistic Regression

Logistic Regression uses maximum likelihood to estimate parameters in the model:

$$\ln\left(\frac{p(Good)}{1-p(Good)}\right) = b_0 + b_1x_1 + b_2x_2 + \cdots + b_kx_k \tag{3.1}$$

As can be seen, the dependent variable is transformed into the log-odds. It is not possible to use an analytical approach to find the parameters of the model (unlike with Linear Regression and OLS), so an iterative process must be used. The process starts with random parameters, which are iteratively modified until the likelihood function (equation 3.7 below) is maximized.

More specifically the Logistic Regression model begins with the assumption that for each potential outcome of the dependent variable, the probability of y = 1 is p, while the probability of y = 0 is (1-p). p is modelled as:

$$p = \frac{\exp(Z)}{1+\exp(Z)} = \frac{1}{1+\exp(-Z)} \tag{3.2}$$

$$Z = b_0 + b_1x_1 + b_2x_2 + \cdots \tag{3.3}$$

This follows from,

$$\Pr(y_i = 1) = \Pr(Z_i + \varepsilon_i \geq 0) = \Pr(\varepsilon_i \geq -Z_i) = \Pr(\varepsilon_i \leq Z_i) \tag{3.4}$$

, with i for each observation in the dataset. ε_i is the part of Z not accounted for by the predictors.

The last probability in the equation above is thus the cumulative distribution function (CDF) evaluated at Z_i. Now, assuming ε_i follows a standard logistic distribution,

$$\Pr(\varepsilon_i \leq Z_i) = \frac{1}{1+\exp(-Z_i)} \tag{3.5}$$

, which follows since the CDF of the logistic distribution is:

$$\mathrm{F}(Z;\mu,s) = \frac{1}{1+\exp\left(-\frac{Z-\mu}{s}\right)} \tag{3.6}$$

, where μ is zero and s is 1, since it is the CDF of the *standard* logistic distribution. The solver then maximizes the log-likelihood function:

$$\ln(\text{likelihood}) = \sum_{i=1}^{n} [y_i \ln(p_i) + (1 - y_i) \ln (1 - p_i)] \tag{3.7}$$

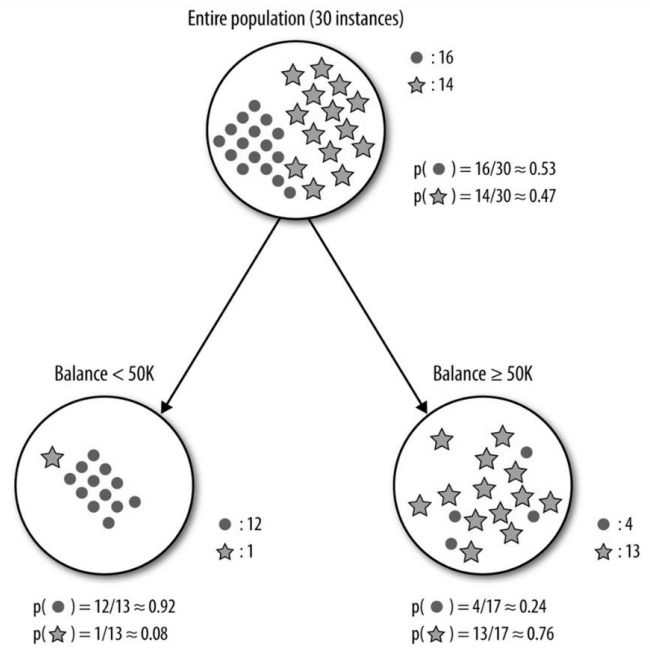
, where for ease of reading equation 3.5 is set to p_i. As can be seen, each predictor affects p_i through Z_i.

Importantly for the modeling of PD, the Logistic Regression does not make the modeler decide on hyperparameters before estimating the model. Thus, it is easy to implement and reproduce.

3.1.2 Decision Trees

Decision Trees split the data into partitions with operations at each branch. The top node is called the root node, and each node underneath is a child node. At the bottom of the tree are leaves, nodes that are either entirely pure or that is not split further due to size constraints. The nodes are split in two or more (except for the leaves) and the depth of the tree, as in the number of levels, is normally part of the input (maximum depth). Figure 3.1 gives an example of a simple decision tree with only one split (Provost & Fawcett, 2015). Intuitively, the child nodes are "purer" than the parent, in that they are more homogenous. This can be seen from the fact that the share of each type is more different in the child nodes than in the parent.

Figure 3.1 Visual illustration of partitioning in Decision Tree



To split the dataset, several splitting rules may be chosen. One common approach is to use an entropy measure to calculate the information gain (IG) of the split, such that

$$IG(parent, children) = entropy(parent) - [p(c_1) \times entropy(c_1) + p(c_2) \times entropy(c_2) + \cdots] \quad (3.8)$$

It is common to begin at the top node and recursively partition the data such that the IG is greatest at each split. This is a type of greedy algorithm, since a local optimum is solved for at each split to try to find the global optimum.

As can be seen, IG measures the difference between the entropy of the parent and the weighted sum of the entropy of the children, where each child is denoted (c_k). Entropy is calculated as:

$$entropy = -p_1 \log(p_1) - p_2 \log(p_2) \quad (3.9)$$

When using Decision Trees, some hyperparameters must be specified. For instance, entropy is not the only purity measure. The Gini index is a common alternative. *Maximum depth* specifies the maximum number of recursive partitions that are allowed, i.e. how deep the tree is. *Maximum branch* specifies the maximum number of branches that may be split in each node.

3.1.3 Neural Networks (NN)

NNs may be visually represented as "neurons" distributed between layers, where the first layer is considered the input layer, and the last layer is considered the output layer. Each neuron takes inputs, computes a weighted sum of the inputs, and then uses an activation function to transform it in a non-linear way (Mueller & Massaron, 2016).

The architecture of a neural network describes the number of neurons and how they are arranged in layers. Typically, one two or three layers are used, with neurons split equally between these layers. The number of layers and neurons are part of the hyperparameters to be specified before training.

NNs are great at recognizing patterns in data. Although they are not much used in practice for PD modelling, other areas within banks are more or less dependent on the models, for example in fraud detection and Anti Money Laundering (AML). Unfortunately, NN is perhaps the

machine learning technique most prone to overfit. This comes as a consequence of its great capacity to recognize patterns. Another issue with NNs in PD modelling is the difficulty of interpreting what drives the model to classify some as good or bad. This black box nature of NNs limits transparency.

3.1.4 Random Forests

Randoms Forest models take advantage of a technique called Bootstrap Aggregation (Bagging for short). With this technique, random samples with replacement from the dataset are chosen with pairs of the feature vector and the dependent variable. Trees are then built on all these samples. Then, in a classification setting, the mode of the predictions for all trees is used for estimating the dependent variables on new observations. This technique will generally increase performance since variance is reduced as long as the correlation between the trees is relatively low. In addition, Random Forests are characterized by the fact that variables chosen to partition the data in each split are chosen at random, not by any measure of information gain, as described in the section on Decision Trees.

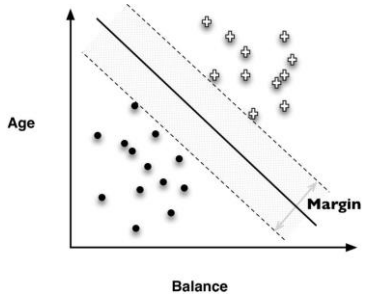
Thus, Random Forest models are differentiated from Decision Trees in two respects: 1. several trees are generated, trained on different samples, and 2. The variables chosen in each node are not chosen based on discriminatory power but simply by random (Krauss, 2014).

In generating any Random Forest model, at least two parameters must be specified; the number of trees to grow (i.e. the number of samples to be selected) and the number of variables to consider at each node.

3.1.5 Support Vector Machines (SVM)

SVMs are classifiers defined by separating the feature space with a hyperplane. With only two features this may be visualized as a line separating the labeled data, as in Figure 3.2.

Figure 3.2: Visual Illustration of SVM with Two Features



With three dimensions it may be visualized as a plane separating the data, but with more dimensions it becomes difficult to visualize. In figure 3 a linear SVM is drawn, but non-linear SVMs also exist, making use of the so-called kernel trick to separate the data (Provost & Fawcett, 2015). Also, it is rare for data to be perfectly linearly separable, as in Figure 3.2. Therefore, in most cases a hinge loss function is used, where data on the wrong side of the separator is "punished" proportional to the distance from the separator. These hinge loss functions normally do not use the square of this distance, unlike ordinary least squares, making SVMs less prone to adjust to outliers. The separator is thus formed by maximizing the margin, defined as the distance between the separator and the nearest data point.

The above discussion sheds light on the three important hyperparameters for the SVM. *Gamma* is a parameter of the kernel for non-linear classification. *Cost* is the cost of misclassification in the hinge loss function. *Epsilon* is the margin of tolerance.

3.1.6 K-Nearest Neighbor (KNN)

The KNN algorithm classifies new observations based on target values for the nearest neighbors in the feature space. What is defined as a near neighbor is part of the hyperparameters. For example, one could define the number of neighbors to be 10. The new observation is then assigned to the class that is most common among the 10 nearest neighbors. Most often, the algorithm is weighted, such that the nearest neighbors have higher weights (Provost & Fawcett, 2015). Also, the "distance" to a neighbor is often defined as the Euclidean distance. The number of neighbors to be chosen which most optimally discriminates between the classes is an empirical matter and different values should be tried.

3.1.7 Gradient Boosting

A Gradient Boosting algorithm seeks to approximate a function of weights on weaker classifiers (such as Decision Trees) to minimize the loss function. The algorithm starts with arbitrary weights and proceeds in a "greedy" fashion. Many Gradient Boosting algorithms are based on the recursive partitioning algorithm described in Friedman (2001) and Friedman (2002). Decision Trees are used as weak classifiers, making use of "Tree boosting". This creates a series of decision trees from samples of the data (SAS, 2017). The hyperparameters that must be specified are reflections of the above description.

Iterations specifies the number of trees to be grown. *Train proportion* specifies the percentage of data to train each tree with. Further, the more general Decision Tree hyperparameters must be specified, such as *Maximum branch* and *Maximum depth*, as described above.

3.2 Weight of Evidence and Information Value

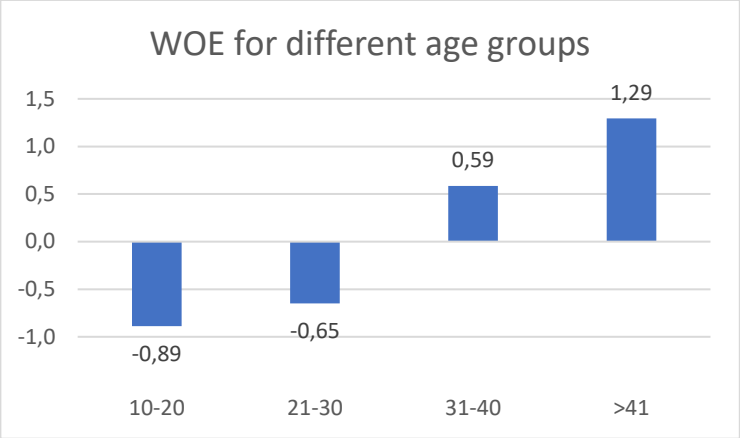
3.2.1 Weight of Evidence (WoE)

In PD modelling, the Logistic Regression is typically combined with Weight of Evidence (WoE) transformations (Anderson, 2007). In this transformation, each explanatory variable is replaced with its WoE. There are several advantages of using WoE transformed variables (Siddiqi, 2006):

- It makes it possible to capture non-linear relationships between the predictor and the dependent variable (explained below)
- It explicitly handles the issue of outliers by either grouping them in an existing bin or creating a new bin
- It handles the issue of missing variables by either grouping them in a separate bin or for example grouping them in the bin with the greatest number of observations
- It allows for a monotonic increase or decrease in the effect of a variable
- It makes the results easy to understand

WoE is used to assess the relative risk of the attributes within a characteristic (variable). Thus, attributes with similar risk characteristics are typically merged. WoE is typically available for calculation in statistical packages, where the output typically is a graph as depicted in Figure 3.3. As can be seen, although the WoE is monotonically increasing with age, it is not increasing linearly. As a result, non-linear effect can be identified.

Figure 3.3: Example WoE Transformation



The WoE of an attribute is calculated as (SAS Institute Inc, 2013):

$$WOE_{attribute} = ln\left(\frac{p_{attribute}^{non-event}}{p_{attribute}^{event}}\right) = ln\left(\frac{\frac{N_{non-event}^{attribute}}{N_{non-event}^{total}}}{\frac{N_{event}^{attribute}}{N_{event}^{total}}}\right) \quad (3.10)$$

$N_{non-event}^{attribute}$ = the number of nonevent observations that exhibit the attribute

$N_{non-event}^{total}$ = the total number of nonevent observations

$N_{event}^{attribute}$ = the number of event observations that exhibit the attribute

N_{event}^{total} = the total number of event records

An example of the calculation of WOE for age is given in Table 3.2. As can be seen, the WoE for the attributes are consistent with the visual illustration in Figure 3.3. Also, the WoE is monotonically increasing, making interpretation easy and intuitive. Often, constraints on the number of bins are part of the software, for example a minimum of 5 percent of the observations in any bin. When optimizing the bins, both the absolute size of WoE for each attribute, and the difference in WoE between each attribute, is maximized. The larger this difference, the greater predictive ability of the characteristic. Missing values are often placed in a separate bin, since assuming that observations with missing values have characteristics that do not systematically vary from non-missing observations is probably wrong. Also, the computer's optimization of bins is part of an iterative process with business judgement driving optimization of bins. For example, in Norway there are regulations on loan-to-value both for

primary and secondary houses. For primary houses, 15 percent equity is required, while for secondary houses, 40 percent is required (Lendo, 2019). Binning should capture this regulatory pattern. Business judgement thus plays a significant role in the credit scoring process, often involving people from operations and different business units.

Table 3.2: Example WoE Calculation

Range	Bin	Non-events	Events	% of non-events	% of events	WOE	IV
10-20	1	589	345	17 %	41 %	-0,887	0,215
21-30	2	601	278	17 %	33 %	-0,651	0,103
31-40	3	938	126	27 %	15 %	0,586	0,070
>41	4	1345	89	39 %	11 %	1,294	0,364
Total:		3473	838				**0,752**

3.2.2 Information Value

A related concept to WoE is Information Value (IV), which is given in the last column of Table 3.2. IV is often used to select predictors in the univariate selection processes. IV is defined as:

$$IV = \sum_{i=1}^{m} \left(\frac{N_{non-event}^{attribute}}{N_{non-event}^{total}} - \frac{N_{event}^{attribute}}{N_{event}^{total}} \right) \times WoE_i \qquad (3.11)$$

, where m is the number of bins. As can be seen, IV is a weighted sum of the WoE of the characteristic's attributes. Thus, the higher the IV, the higher is the predictive ability of the characteristic. In the industry, but also in textbooks, rules of thumb are used to describe the IV of characteristics. According to Siddiqi (2006), one rule of thumb is that IV of:

- Less than 0.02 is regarded as unpredictive
- From 0.02 to 0.1 is regarded as a weak predictor
- From 0.1 to 0.3 is a medium predictor
- Greater than 0.3 is a strong predictor

It is important to note that IV increases with number of bins. Thus, one should balance the ease of interpretation when there are few bins with greater discriminatory power for more bins.

Of course, univariate analysis is not enough for choosing which variables should enter a regression. Typically, all variables that pass the univariate analysis enter a stepwise regression

procedure. You then (hopefully) end up with a few variables that have both high individual predictive power and that have low correlation.

3.2.3 Akaike's Information Criterion

The Akaike's Information Criterion (AIC) is a statistic commonly used in the stepwise regression procedure. When comparing models with different AICs, the model with the lowest AIC is chosen, all else equal. The formula for computing the statistic is given below (Konishi & Kitagawa, 2008):

$$AIC = 2k - 2\ln(L) \tag{3.12}$$

As can be inferred from the definition, the statistic penalizes models with many parameters (k) and gives preference for models with high model fit, measured by the likelihood function (L). As such, the statistic may be interpreted as a tool to select models while avoiding overfitting.

3.3 Evaluation Methods

Several different metrics may be used to evaluate performance of a credit scoring model, and the most common are the Gini statistic, the Receiver Operating Statistic Area Under the Curve (ROC AUC) and the Kolmogorov-Smirnov (KS) statistic. The KS statistic is a very simplistic measure. And, since ROC AUC is approximately a linear function of the Gini statistic (equation 3.13), only the former metric is used.

$$ROCAUC \approx (1 + Gini)/2 \tag{3.13}$$

To explain the ROC AUC statistic, two concepts must be explained first, *sensitivity* and *specificity:*

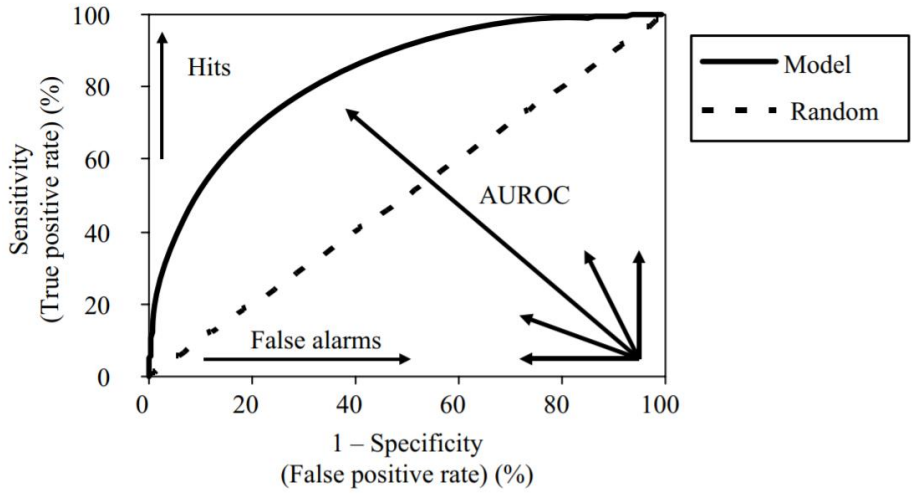
$$Sensitivity = \frac{True\ positives}{True\ positivies + False\ negatives} \tag{3.14}$$

$$Specificity = \frac{True\ negatives}{True\ negatives + False\ positives} \tag{3.15}$$

Sensitivity measures the ability to detect true positives, while specificity measures the ability to detect true negatives. There is an inherent tradeoff between the two. As such, a graph can

be made with 1-specificity on the x-axis and sensitivity on the y-axis. This gives the Receiver Operating Characteristic (ROC) curve, as seen in Figure 3.4. The figure illustrates that a model randomly assigning default or non-default to the observations is a straight line in the ROC chart. In general, the goal for the model is to maximize the area under the ROC, i.e. having as high sensitivity as possible for the same false positive rate. This area is abbreviated AUC – Area under the curve, thus the name of the metric is ROC AUC.

Figure 3.4: Illustration of the ROC (Anderson, 2007)



The KS statistic measures a slightly different aspect of performance. It is built on the comparison of the empirical cumulative distribution function (ECDF) and some other CDF, in our case another ECDF. For PD modelling, the first ECDF is the distribution of bads, while the second ECDF is the distribution of goods. The KS statistic is then defined as the maximum difference between the two CDF's. Again, this is a very simplistic measure, since it only considers the difference at a given point (not the entire area).

3.4 Hyperparmeter Tuning

Hyperparameter tuning is the term used for the process of finding the optimal values for hyperparameters in machine learning algorithms. These hyperparameters are different from other parameters in that they are set before learning, while the other parameters are learned. The "optimal values" can be described as the set of hyperparameter values that minimize some pre-defined loss function.

Often, a Grid Search is performed to find the optimal hyperparameters. A grid search is performed as an exhaustive search on a set of different values for the hyperparameters, which are usually chosen beforehand. Of course, since there might be a huge number of combinations for the hyperparameters, this is a high dimensional problem.

Random Search, another method for hyperparameter tuning, takes advantage of the fact that often there are only a few hyperparameters that determine the performance of the model. By randomly selecting values for hyperparameters, the aim is to get as close as possible to the best solution found with a Grid Search.

4. Data Description

4.1 Data Source and Anonymity

The dataset has been collected from a financial services firm. It was created by extracting information from the firm's data warehouse. There are 891 554 observations, which consist of all accepted mortgage agreements from 2009 to 2017. Since the observations describe mortgage agreements at the time of application, the dataset is characterized as an "application" dataset, unlike a "behavioural" dataset, which follows the individuals over time.

The financial services firm shall remain nameless throughout this thesis. With the information provided in this thesis competitors might infer how well the models of the firm in question work, or stakeholders might obtain information that should only be held by company insiders. Henceforth the financial services firm shall be referred to as "the firm".

4.2 Programming Language

Since the firm did not want the dataset to be exported to Python or R, all operations were done in the firm's internal system, SAS Miner. SAS Miner is a popular program for building credit scoring models in the industry. Although the program is not much used in academia, there is no reason to believe that using another program would be better.

4.3 Candidate Variables

There are 549 candidate variables. These were previously identified through discussions and workshops by the employees of the firm, both within the department making the model and between product departments. The variables are related to risks that the firm has deemed potentially important for the behaviour of loan applicants. The variables may be placed in the following categories, with several variables in each category:

- Transactions related to the customer, including transactions related to the checking and savings account
- Measures of the size of the loan in relation to equity and income
- The obligor's savings and debt

- Characteristics related to the obligor, such as age, marital status etc.
- Characteristics related to the co-debtor
- The obligor's previous credit history, including payment remarks and reminders

4.4 Target Selection

There exists one observation per obligor, and for this observation, there are 24 binary variables for each month following the beginning of the agreement, where a value of "Yes" means that the customer is at least 90 days due with a payment. A new variable is created to define default, "Default_24". This is a binary variable equal to unity if there is "Yes" in any month variable and the amount due is at least 500 NOK. This is in line with the discussion from the literature review on Basel requirements.

There are two considerations that had to be made when defining a default. First, what is the appropriate number of months to include in the prediction? Industry practice is to estimate default for either 12 or 24 months following the loan agreement. A default period of 24 months is used in this thesis to make sure that long-term defaults are captured, not only short-term. The second consideration is what the size of amount due should be set at. The Basel regulatory framework does not set an exact limit for this and instruct banks to act on its own judgement (Basel, 2005). 500 NOK seems like a likely significant amount. Anyhow, the cut-off would not influence the expected loss calculation, since a higher cut-off would yield lower PD, but higher Loss Given Default (LGD).

4.5 Data Partitioning

The data is partitioned into a training, validation and test split. The training split is used to train the model and obtain the best model weights. The validation split is then used to fine-tune the model to avoid overfitting. The test split is used solely to examine the predictive abilities of the model, for example to test the ROC AUC statistics. For small datasets one should consider not including the test split, as it reduces the number of observations available for training, but since our dataset consist of almost one million observations it is deemed appropriate to use all three splits (SAS, 2017). The observations are sampled into each of the three splits with a random number generator (seed set at 12345)

The standard training, validation and test split of 40-30-30 in SAS is used. Anyhow, due to the large amount of observations this is not deemed important.

4.6 Oversampling

Due to the nature of mortgage defaults and the strict regulation surrounding it, with protection of consumers, defaults are very rare in the sample. For the firm to remain anonymous, the exact percentage shall not be disclosed. For datasets involving PD modelling it is common practice to oversample the dataset (SAS, 2017). For large datasets this can tremendously decrease model fitting time, which is highly appreciated when many models are built. The dataset it therefore concentrated to a 2 percent event rate.

4.7 Deflation of Monetary Values

Several variables in the dataset are monetary, and originally registered nominally. This of course creates a problem in that changes in the general price and salary level makes the model weights wrong if not adjusted. Monetary values are therefore deflated to 2005 level using the SSB consumer price index (SSB, 2019), except for income variables, where the SSB salary index is more accurate (SSB, 2019). The following formula has been used to deflate variables:

$$X_{it}^{r} = \frac{X_{it}^{nom}}{P_t^{t_0}} \tag{4.1}$$

, where X_{it}^{r} is the deflated value of variable i in period t and $P_t^{t_0}$ is the Consumer Price Index in period t.

4.8 Reject Inference

The model is built on a dataset with known good and bad customers. That is, applicants that were actually given a loan. This creates a sample bias, since through-the-door customers do not have the same characteristics as the dataset customers unless the previous loans were issued completely randomly, which is highly unlikely. Therefore, a method is required to account for this. *Reject inference* is a term used for these methods.

The sample bias creates problems not only with credit scoring new applicants, but also from a performance and policy perspective. The sample bias may give an artificially low PD level for the portfolio, which decreases the expected and unexpected loss and thus the regulatory capital. In cases with very high approval rates, reject inference becomes less important, as the assumption that all rejects equal bad customers is almost true. This is close to reality in mortgage lending, as lenders typically do not approve loans on the basis of expected loss prediction, but on whether the customer fulfills certain criteria regarding loan-to-value, debt ratio, etc. In such an environment, it is not recommended to use reject inference methods (Siddiqi, 2006). Therefore, such methods are not considered in this thesis. Further, the use of such methods may be very expensive. There are no techniques to estimate with certainty the characteristic of rejected applicants, as they have been rejected. The closest one could get to an experiment is to randomly accept applicants for some period of time, but this would be an expensive experiment to run, with few financial benefits in the case of mortgage lending.

4.9 Variable Definitions

Table 4.1 displays the variables in the dataset, grouped in variable categories. For variable categories with very similar variables not all are represented in the table below. Also, due to the number of variables in the dataset, only variables that are not obviously defined (such as age) *and* are relevant for the coming chapters are mentioned in the subchapters below. Other variables, such as the employment variable, are described in appendix 1.

Table 4.1: Overview of Variables in Dataset

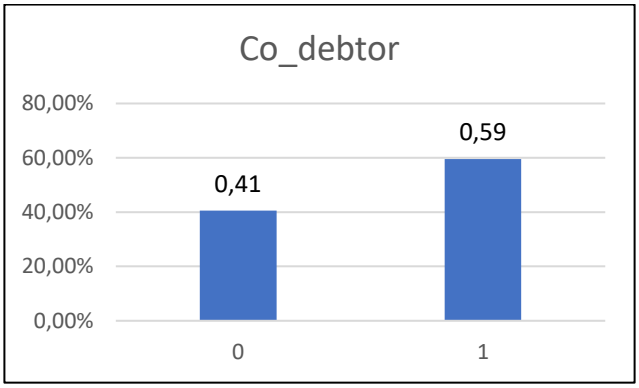
Variable category	Variable	Definition
Customer characteristics	Age	Age of debtor
	Housing	Whether obligor is living in a house, apartment, etc.
	Employment	Employment condition of debtor
	Children	Number of children of debtor
	Co_debtor	Binary variable on whether debtor has co-debtor
	Relationship	Relationship status of debtor
	Mortgage	Binary variable on whether debtor has had a mortgage with the firm in the last 12 months
	Other_loan	Binary variable on whether debtor has had other financing with the firm in the last 12 months (for example for a car loan)
Transaction variables	Fixed_in_trans_MIN12	Sum of fixed incoming transactions, minimum in the last 12 months
	Fixed_num_out_trans_STD12	Number of outgoing transactions, standard deviation last 12 months
Credit card variables	CC_limit_MIN6	Limit utilization of credit card, minimum in last 6 months
Savings variables	Savings_balance_AVG3	Balance on savings account, average last three months
Payment reminder variables	Pay_rem_num_24m	Number of first-time and second-time payment reminders received in the last 24 months
	Pay_rem_last	Days since last payment reminder
Default variables	Previous_defaults_all	Describing if obligor has defaulted on previous agreements
Payment remark variables	Payment_remark	Variable indicating if the obligor has a payment remark
Loan/economic characteristics	Net_income	Sum of the debtor's and co-debtor's net income
	Debt_to_equity	Debt to equity
	Debt_to_income	Debt to income, including co-debtor
	Net_worth	Net worth of main debtor and co-debtor
	Granted_loan	Absolute size of loan

4.9.1 Co-debtor Variables

There are several variables that describe the co-debtor. However, there is one specific variable describing whether the main-debtor has a co-debtor at all. As we shall see later, this is

important when designing interaction variables. Co_debtor is a binary variable equal to unity if the applicant has a co-debtor, with a mean of 59 percent, as is seen from figure 4.2. There are no missing values for this variable. The following frequencies are observed:

Figure 4.1: Distribution of Values in Co-debtor Variable



4.9.2 Transaction Variables

There are several transaction variables. There are variables concerning whether the transaction has gone into the account or out of the account, variables measuring the sum of all transactions or the number of specific transactions. Also, there are variables describing both variable transactions and fixed transactions (for example leasing costs for a car every month).

In addition, several statistics are computed, significantly increasing the number of variables. The following list is an exhaustive account of all statistics considered:

- MIN: Minimum
- MAX: Maximum
- AVG: Average
- MED: Median
- SUM: Sum
- STD: Standard deviation

Further, after each statistic, the number of months prior to the application is considered. For example, MIN3 would be the minimum for the last three months. Putting it all together, an example variable name would be the 12-month standard deviation of the sum of variable transactions going out of the account.

As one would expect, all these options result in many variables in this category. In total there are 181 transaction variables.

4.9.3 Savings Variables

The savings variables contain statistics on the balance of the savings account of the applicant, including the balance on any stock market funds or other investment vehicles the applicant may be invested in. This is viewed as a reasonable way to measure savings, since most customers not only invest in their low interest rate savings account but also in higher yielding funds.

These variables are measured with the statistics MIN, MAX, AVG, MED, SUM or STD, as defined above. Of course, not all applicants have a savings account in the firm at the time of application, giving some missing values (19 percent for all variables). In total there are 31 savings variables.

4.9.4 Credit Card Variables

For applicants with credit cards which have been used prior to the application date there should be useful information regarding their repayment behavior and thus their creditworthiness.

There are three types of credit card variables, measuring:

- Number of transactions
- Percent of limit drawn
- Size of the interest expense balance

As with previous variables the credit card variables are measured on several statistics. Not all applicants have credit card history in the firm, giving 30 percent missing values.

4.9.5 Payment Reminder Variables

There are three payment reminder variables in the dataset, “Pay_rem_num_24m”, “Pay_rem_last” and “Pay_rem_last_co”

“Pay_rem_num_24m” is an interval variable measuring the number of first-time and second-time payment reminders the obligor has received in the last 24 months.

"Pay_rem_last" describes the number of days since the last payment reminder for the main obligor. There is a corresponding variable for the co-debtor, which is also included in the dataset, "Pay_rem_last_co". A missing value for the Pay_rem_last variables indicates that the main or co -debtor have not received any payment reminders. For the number of payment reminders variable, we should separate between customers that have the value 0 because they have not defaulted, and those that have the value 0 simply because they are a new customer. This extra logic is introduced later.

4.9.6 Default Variables

Default variables describing previous defaults are separated in categories to indicate whether the variable is based on credit card agreements, mortgage agreements, other agreements or all. Previous defaults are calculated based on either the last 12 or 24 months. The variables may take on either "Missing", "Small_def" or "Large_def" as value. The co-debtor also has similar variables. Customers without registered defaults have Missing as value, while Small_def and Large_def denotes less serious and serious defaults respectively. A less serious default is defined as a default where the customer has paid interest but not principal to delay repayment, while in a serious default neither interest nor principal has been paid.

To make sure that customers with an earlier agreement with the firm who has not defaulted are separated from those that simply has not had an agreement with the firm (and therefore not defaulted), additional logic is needed, as is described later.

4.9.7 Payment Remark Variables

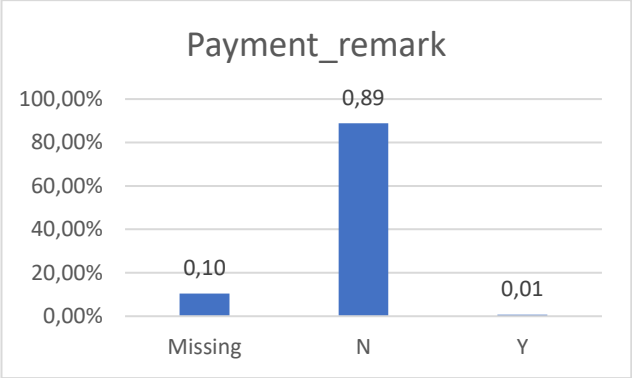
Payment remark variables are calculated in different categories, measuring either the number of payment remarks or the size of the payment remarks. This could be payment remarks for the main obligor (the applicant) or payment remarks for the co-debtor.

The variables used in the final model may take on any of the following three values:

- Missing: Missing information on this application
- N: There is no payment remark at the time of application
- Y: There is a payment remark at the time of application

Frequencies for Payment_remark is seen in Figure 4.4.

Figure 4.2: Distribution of Values in Payment Remarks



4.9.8 Loan/Economic Characteristics

Debt-to-Equity

The Debt-to-equity variable measures the size of the loan relative to equity. Since there is regulation on minimum equity contribution at 15 percent and 40 percent, the weight of evidence variables which are outlined in the next subchapter prove especially useful, as exact cutoffs around 15% and 40% can be made. More information is therefore provided in the chapter on weight of evidence transformations.

Debt-to-Income

The Debt-to-income variable is defined as the sum of debt for both debtor and co-debtor divided by the sum of gross income for both debtor and co-debtor. Again, because of regulatory limits it is helpful to discretize this variable. This will be elaborated upon in the chapter on weight of evidence transformations.

5. Interactions and Selection of Variables

5.1 Variable Interactions

As part of the exploratory analysis of a dataset it is common practice to transform variables to increase explanatory power. The two main ways of transforming variables are with polynomials/logarithms and with interactions. Transformations with polynomials or logarithms are common when the objective is to better fit non-linear patterns in the data. However, since the weight of evidence transformations account for these non-linearities, the purpose of doing polynomial/logarithmic transformations is gone. Still, the weight of evidence transformations does not account for interactions that might exists between variables.

Table 5.1 displays variable interactions performed, and the inputs to the interaction variables. Generally, we want to combine variables with information on whether there has actually been a customer relationship. For example, a variable that only can take on the values “has defaulted with the firm in the past” and “has not defaulted” is less informative than if the variable could take on the values “has defaulted with the firm in the past”, “has not defaulted *and* has been a customer with the firm for 24 months before application” and “has not defaulted *and* has not been a customer with the firm prior to the application”. Indeed, if the customer has not defaulted *and* has been a customer with the firm, she has proven to be responsible. This is just one example of several variables where it could be appropriate to interact a variable with another variable on the customer’s relationship with the firm. As can be seen from table 5.1, this interaction has been performed for several variables, i.e. whether the customer has had a mortgage or another loan in the past has been accounted for.

Table 5.1: Illustration of Interactions

Interaction variable name	Inputs to interaction variable
Previous_defaults_all_customer	Previous_defaults_all
	Mortgage
	Other_loan
Co_debtor_default	Co_debtor
	Co_debtor_previous_def = max(Previous_defaults_mort24_co, Previous_defaults_CC24_co , Previous_defaults_other24_co)
Pay_rem_num_24m_customer	Pay_rem_num_24m
	Mortgage
	Other_loan
Pay_rem_last_customer	Pay_rem_last
	Mortgage
	Other_loan
Payment_remark_both	Payment_remark
	Payment_remark_co
Debt_to_income	Debt_to_income
CC_limit_AVG12_customer	CC_limit_AVG12
	CC_quantity_SUM12

5.1.1 Interactions on Default Variables

Previous_defaults_all was earlier introduced as a default variable. Unfortunately, this variable did not separate new customers to the firm from those that already have a relationship. The new, modified version can take on any of *four* values. This can be done because it is interacted with two variables describing whether the customer had a credit obligation with the firm in the past. The new values are given below:

- Missing: No default
- New_customer: This is a new customer to the firm, thus there are no previous agreements
- Small_def: Minor default in one or more agreements in the past

- Large_def: Serious default in one or more agreements in the past

To qualify as missing (i.e. no default) the customer must have had an agreement with the firm 12 months prior to the application. If not, it is registered as “New_customer” (if there are no defaults). This variable is thus able to separate those customers that have “proven” to be low risk borrowers from those that have not proven this.

5.1.2 Interactions on Co-Debtor Default Variables

There are two variables entering this interaction – Co_debtor and Co_debtor_previous_def. The last variable provides information on the historical defaults of the co-debtor, and considers mortgage loans, credit card loans and other loans. Co_debtor is a binary variable with information on whether the main-debtor has a co-debtor.
Co_debtor_default is the interaction of these two variables. This nominal variable is more complex than others in that it provides information from two sources. First it separates those that do have a co-debtor from those that do not. Secondly, it contains information on possible defaults by the co-debtor in the period before the mortgage application. The variable divides customers into five categories:

- Co-debtor with no defaults: The applicant has a co-debtor and he/she has not defaulted in the past
- Co-debtor new customer: The customer has a co-debtor, but he/she is new to the firm and thus there is no information on past payment behavior
- No co-debtor: The applicant does not have a co-debtor
- Co-debtor with small default: The applicant has a co-debtor and he/she is registered with a minor default in the past
- Co-debtor with serous default: The applicant has a co-debtor and he/she is registered with a serious default in the past

All credit obligations the co-debtor may have had in the past are considered, including mortgages and credit cards. There are no missing values for this variable.

5.1.3 Interaction on Payment Reminder Variables

Pay_rem_num_24m_customer is created by the interaction between Pay_rem_num_24m, Mortgage and Other_loan, as can be seen from Table 5.1.

Due to difficulties in interpreting missing values (could be either an unknown customer to the firm, or a good customer), extra logic is needed. A new value, -999, is given to customers without a credit obligation in the past. This extra logic makes the variable significantly better at discriminating good and bad applicants.

Pay_rem_last_customer is created by the interaction between Pay_rem_last, Mortgage and Other_loan. There is a similar interaction for the co-debtor. For these variables, missing is the best value, since that means no payment reminders have been received. Still, additional logic is needed for this variable, since applicants without any credit obligations with the firm should not receive the same score as applicants with previous obligations to the firm and without any payment reminders, since applicants in the latter category have "proven" their creditworthiness. Customers without a credit obligation in the past is given the value -999, such that observations with missing value both have had a credit obligation *and* did not default.

5.1.4 Interaction on Payment Remark Variables

Since the firm has info on both debtor and co-debtor payment remarks, it could be helpful to create a new variable which takes both into account. This is what Payment_remark_borth does. The variable is created by combining Payment_remark and Payment_remark_co. We thus get the values:

- N
- Y
- N&N
- N&Y
- Y&Y
- Y&N
- Missing

, where the first letter describes whether the main-debtor has a payment remark (Y) or not (N), and the same for the co-debtor with the second letter.

5.1.5 Interactions on Debt-to-Income Variables

Debt_to_income is already included in the dataset as a variable that considers the debt-to-income ratio of both debtor and co-debtor, i.e. the sum of debt for both divided by the sum of gross income for both. It is therefore not necessary to do anything with the individual debt-to-income variables.

5.1.6 Interactions on Credit Card Variables

CC_limit_AVG12_customer is an interaction between the CC_limit_AVG12 and CC_quantity_SUM12 variables. CC_limit_AVG12_customer takes on the value of CC_limit_AVG12, i.e. the average credit card limit utilization in the past 12 months, but only given that it has been used, i.e. CC_quantity_SUM12 is greater than 0. This should provide some additional information, since it is important to separate between customers that have a credit card that is regularly used *and* have the discipline to repay the debt, and customers that have a credit card but simply do not use it, as they have not proven this discipline.

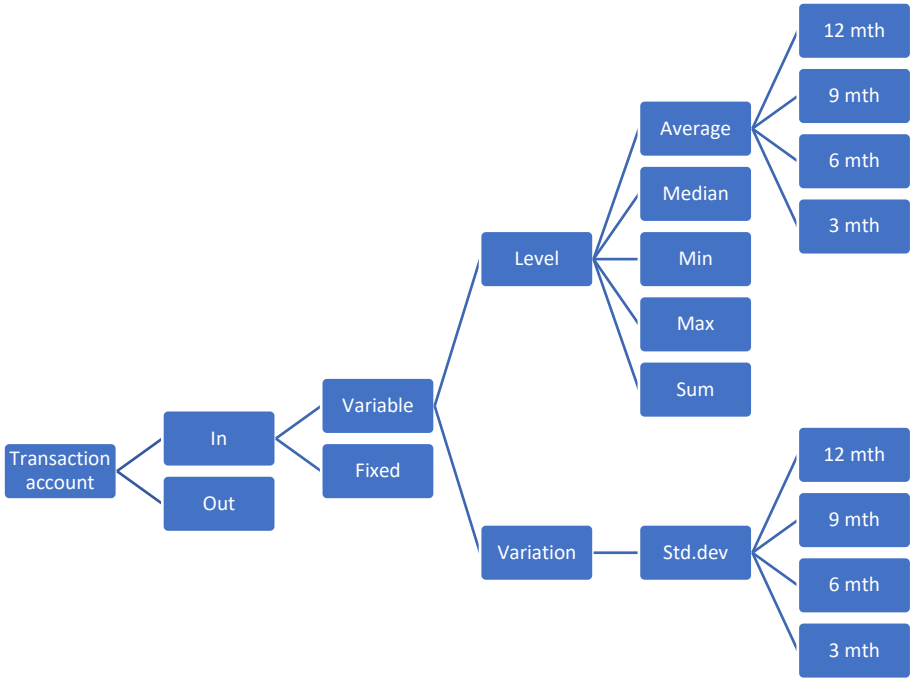
5.2 Variable selection

The variable selection process consists of two parts. In the first part, univariate analysis is carried out on all variables to find variables with high discriminatory power. To compare variables' univariate discriminatory power the Information Value (IV) statistic is used. This provides a quick way to filter out variables that do not have explanatory power. A 0.10 IV cutoff is used as the minimum IV to include a variable for further analysis. This is regarded as conservative, as explained in chapter 3.2.2.

The transaction and savings account variables differ from the other variables in that there are many variations of them. The variables are organized in a hierarchy according to Figure 5.1, which breaks up one of the branches for the transaction variables. A similar figure could be made for the savings account variables. As can be seen, all variables are given with different durations; past 12, 9, 6 or 3 months. To ease interpretability and avoid overfitting, only variables for the last 12 months are included for the transaction and savings account variables.

This is supported by the fact that for most variables, the 12 month statistic has the best discriminatory power, as can be seen in appendix 2. The rest of the variables with IV above 0.10 are included in the Logistic Regression.

Figure 5.1: Overview of Transaction Variables



As discussed above, only variables with IV above 0,10 are included. However, one should not only measure the univariate discriminatory power to decide if a variable should be included in the regression. In the final Logistic Regression, we want variables with both high discriminatory power *and* low correlation with other variables. Indeed, if we include two variables with high discriminatory power but that measure approximately the same thing, it suffices to include one of them.

We cannot know which combination of variables will produce the best Logistic Regression model from the previous discussions on univariate discriminatory power. An iterative selection method is needed to produce the best combination of variables (or an approximation of it). To this end, stepwise regression is carried out, with the criterion being Akaike's Information Criterion (AIC).

After stepwise regression has been performed, we are left with the 19 variables in Table 5.2:

Table 5.2: List of Variables chosen with Stepwise Regression

Variable name	Variable definition
Net_worth	Net worth both main debtor and co debtor
Fixed_in_trans_MIN12	Minumum fixed payments coming into checking account last 12 months
Fixed_in_trans_STD12	Standard deviation of fixed payments coming into checking account last 12 months
Fixed_out_trans _MAX12	Maximum fixed payments coming out of checking account last 12 months
Fixed_out_trans _STD12	Standard deviation of fixed payments coming out of checking account last 12 months
Variable_out_trans _STD12	Standard deviation of variable payments coming out of checking account last 12 months
CC_limit_AVG12_customer	Credit card money drawn as percentage of limit
Net_income	Sum of main and co -debtor net income
Savings_balance _MIN12	Minimum balance on savings accounts last 12 months
Savings_balance_STD12	Standard deviation of balance on savings accounts last 12 months
Employment	Employment condition of debtor
Payment_remark_both	Payment remarks for main and co -debtor
Debt_to_equity	Debt to equity ratio for main and co -debtor
Pay_rem_last_customer	Days since last first time payment reminder
Debt_to_income	Debt to income ratio using sum of main and co -debtor income and debt
Co_debtor_default	Credit defaults of co-debtor
Pay_rem_num_24m_customer	Number of first-time and second-time payment reminders last 2 years
Previous_defaults_all_customer	Default status on any credit obligations with the firm
Granted_loan	Absolute size of the loan

6. Weight of Evidence Transformations

6.1 Weight of Evidence Transformations

When the interactive grouping performs univariate analysis on each variable to find which has IV above 0.10, it first transforms the variables to weight of evidence variables. As explained in chapter 3.2.1, the grouping in WoE variables is done by minimizing entropy. However, the WoE variables should also be adjusted to reflect business practice. In this regard, two variables are particularly important – the debt to income and debt to equity variables. As explained, the values for these variables are regulated by the financial authorities. In particular, debt to income must be below 5, and debt to total value below 85 percent, equal to an equity share of 15 percent. Also, for secondary mortgages, debt to total value must be below 60 percent. Consequently, the WoE variables are adjusted to reflect these business considerations, as can be seen for the respective variables below.

In the WoE transformation the variables are grouped (and discretized for interval variables) according to the figures below. These are then used as input variables for the Logistic Regression and all machine learning techniques.

As can be seen from chapter 3.2.1, the bins in the WoE transformed variables are usually defined. However, for the firm to remain anonymous, this shall not be disclosed. Instead, more general names are used for the bins, such as bin 1, 2 and 3, or low, medium, high.

6.1.1 Transaction Variables

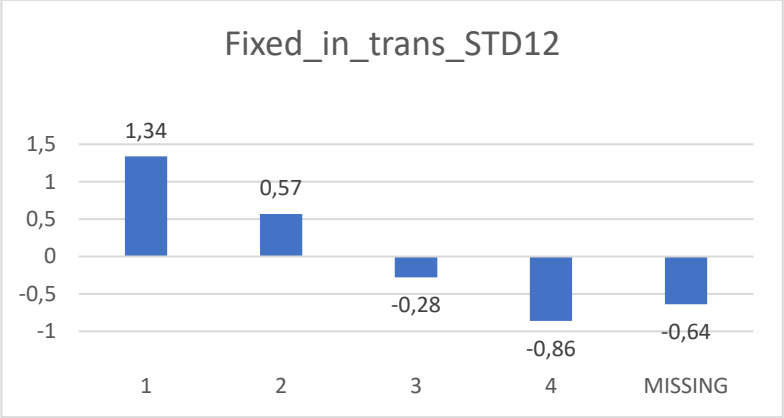
In Figure 6.1, the WoE for each grouping is given for one of the transaction variables. As can be seen, there is in a monotonic decrease in the WoE for this variable - the standard deviation of fixed cash inflows. For example, we can see that the WoE for values in the first bucket is approximately 1,3. The WOE for values higher than this is lower, and each consecutive group has a lower WOE than the one before. As explained in the theory chapter, this is generally preferred when using WOE transformations, as it makes for an easy to understand relationship between the variable and the outcome.

Of course, the Missing category is not part of this monotonic relationship. Anyhow, it is useful to discuss cases where the Missing category has a very different WOE value than expected. In

this case, obligors with a Missing value for the transaction variables do not have any payment history.

There are in total five transaction variables after stepwise regression, but Figure 6.1 is representative.

Figure 6.1: WoE for the Attributes of a Transaction Variable

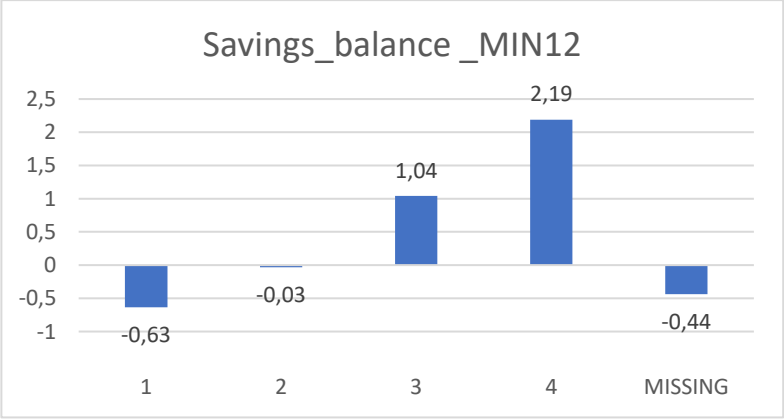


6.1.2 Savings Variables

Figure 7 illustrates the WoE for the discretization of a savings variable, specifically the minimum balance on the savings accounts in the last 12 months. As expected, a higher minimum balance is associated with a higher WOE score. The customers in the Missing category do not have a savings account, and the WOE score for this group is negative.

There are two savings variables passing stepwise regression, but Figure 6.2 is representative.

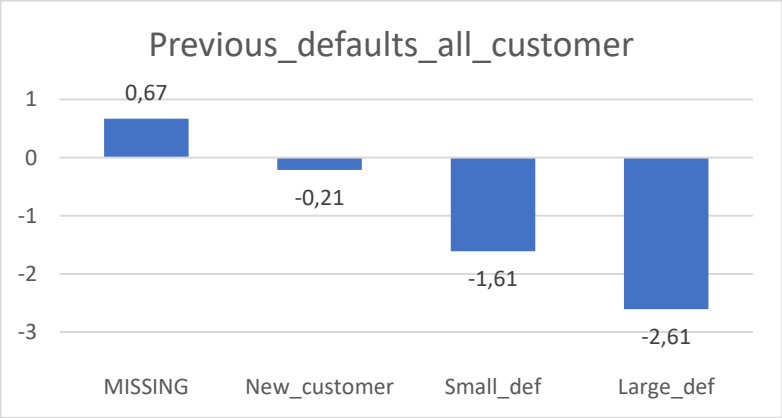
Figure 6.2: WoE for the Attributes of a Savings Variable



6.1.3 Default Variable

The WOE scores for the default variable Previous_defaults_all_customer is seen in Figure 6.3. This variable takes the value of the "worst" default in the last twelve months. As expected, missing is the category with highest WOE, since this means no default. Interestingly, the category "New_customer" has a negative WOE. Intuitively this may be interpreted as that customers with no agreements for the last 12 months have not proven their creditworthiness to the same extent as those in the Missing category, which have had an agreement. Also as expected, those with the value "Large_def", which is the worst default category, have the lowest WOE.

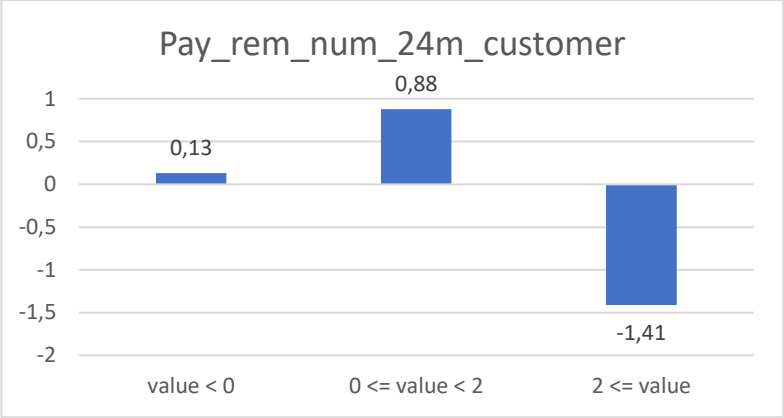
Figure 6.3: WoE for the Attributes of a Default Variable



6.1.4 Payment Reminder Variables

Figure 6.4 displays the WoE transformations for one of the two payment reminder variables. The figure illustrates the relationship that more payment remarks give a lower WoE score. The first category displays customers that have a value below 0, which in all cases is -999, indicating that they have not had an agreement 12 months prior to being scored. That is because the variable consists of an interaction between several variables, as explained in chapter 4.9.

Figure 6.4: WoE for the Attributes of a Payment Reminder Variable

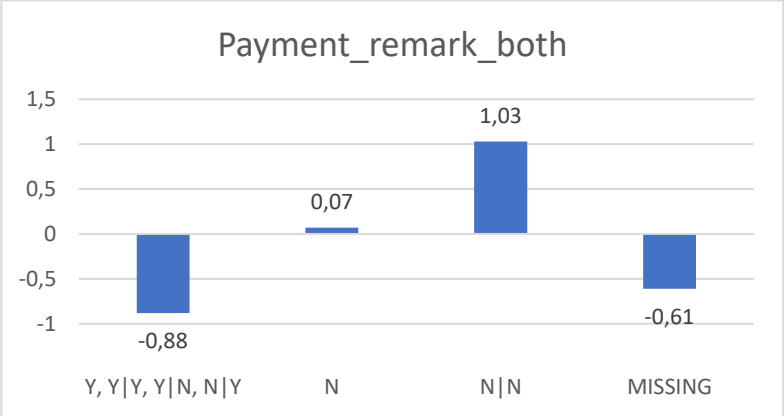


6.1.5 Co-Debtor Variables and Payment Remarks

As explained in chapter 5.1, the payment remark variable is an interaction variable. It considers payment remarks both for the main debtor and the co-debtor. One of the variables going into this interaction is the variable describing whether the main debtor has a co-debtor.

The payment remark variable is seen in Figure 6.5. The groupings are made up of combinations of payment remarks for the main debtor and the co-debtor. For example, N|Y means that the main debtor has no payment remarks at the time of application (N), while the co-debtor has (Y). As can be seen from Figure 6.5, all combinations that contain Y – i.e. that the debtor or co-debtor has a payment remark, are grouped together and receive the lowest WoE score, which makes sense. The highest score is given to those with no payment remarks (N|N).

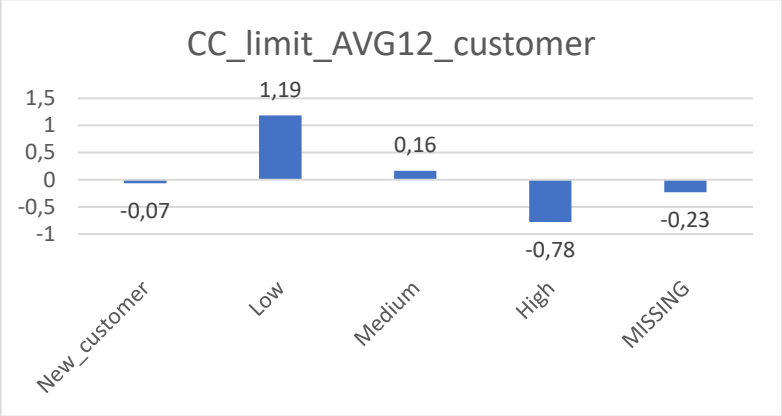
Figure 6.5: WoE for the Attributes of a Payment Remark Variable



6.1.6 Credit Card Variable

Figure 6.6 displays the WOE scores for the credit card variable. This variable describes how many percent of the limitation on the credit card the debtor has drawn. As expected, customers with higher limit utilization are associated with higher default rates, which is why they are given lower WoE score in the transformation. Again, customers in the Missing category are penalized, which intuitively is because they have not proven their credit worthiness. As explained in the variable definitions chapter, customers in the Missing category have had no agreements with the firm in the last twelve months.

Figure 6.6: WoE for the Attributes of a Credit Card Variable

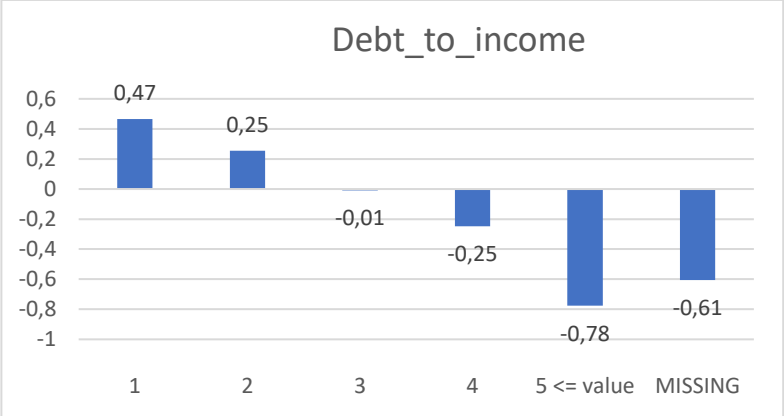


6.1.7 Debt-to-Income Variable

As seen in Figure 6.7, customers with higher debt-to-income ratio are given a lower WoE score. More precisely, it is debtor co-debtor pairs with high debt-to-income that receive low WoE scores, since the variable measures the combined debt-to-income ratio of debtor and co-debtor. As seen in Figure 6.7, there is an exact cut-off for debt-to-income above 5 (other bins are not defined, for anonymity). The WoE transformation performed by SAS had a slightly different cut-off, at 5.04. Since there is a regulatory cut-off exactly at 5, the limits are manually adjusted to this level, to ease interpretation and such that the risk profiles of loan applicant are more accurately reflected. Anyhow, the IV is exactly the same.

It is not clear why some values are missing, since information on income and the loan amount must be gathered for all customers.

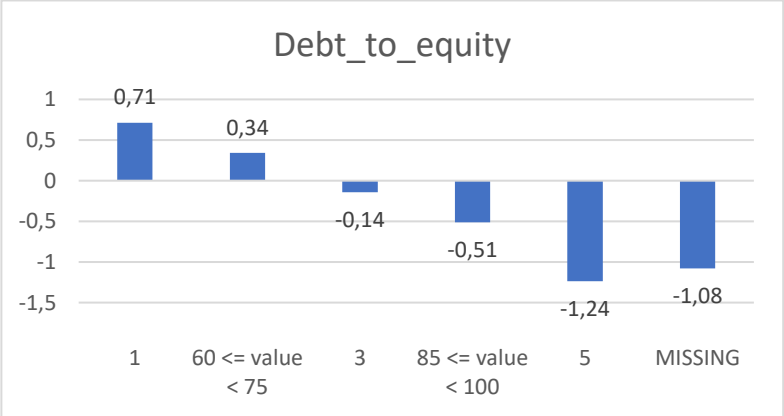
Figure 6.7: WoE for the Attributes of Debt-to-Income Variable



6.1.8 Debt-to-Equity Variable

The WoE transformation for the debt-to-equity variable is seen in Figure 6.8. As expected, higher debt-to-equity ratio gives a lower WOE score, which makes sense because all else equal, these customers have a higher probability of not meeting their obligations. As with the debt-to-income variable, there are regulatory cut-offs at 60 and 85 percent. Therefore, the limits are manually adjusted to account for this. Again, the IV is practically the same, but interpretation and actual business practice is now better reflected in the variable.

Figure 6.8: WoE for the Attributes of Debt-to-Equity Variable



7. Empirical Results

7.1 Hyperparameter Tuning

The machine learning techniques considered in this paper all require some hyperparameter tuning for maximal performance. Since there are typically many parameters to tune for each technique, this is a high-dimensional problem, and manual trial and error would take a long time. Therefore, this is carried out with a grid search optimizer in practice. This is also the method followed here. Table 7.1 displays the hyperparameters that were tuned for each model, and the search space for these hyperparameters. For each machine learning technique there are even more hyperparameters that may be tuned than those presented in the table, but the most important hyperparameters are presented. A full list of hyperparameters for each model is seen in appendix 3 for reproducibility of results.

Table 7.1 Overview of Search Space for Hyperparameter Tuning

Machine Learning Technique	**Hyperparameters**	**Search Space**
Neural Network	Number of layers	1, 2, 3
	Number of hidden neurons	10, 20, 40, 80
	Number of iterations	300, 1000
Gradient Boosting	Maximum branch	30, 60
	Maximum depth	20, 40
	N iterations	100, 200
	Train proportion	30, 60
Random Forest	Maximum number of trees	100, 200
	Maximum depth	30, 50
	Proportion of obs. in each sample	0.3, 0.6
K-NN	Number of neighbors	8, 16
	Weighted	Yes, No
Support Vector Machine	Gamma	1
	Cost	1, 3
	Epsilon	1.0E-6, 1.0E-5
Decision Tree	Maximum branch	10, 20
	Maximum depth	30, 50
	Nominal target criterion	Entropy, ProbChisq

7.2 ROC Charts and ROC AUC Statistics

The NN is a complex machine learning algorithm with many different architectures. A NN could be “deep” in that it has many hidden layers. The NNs with one and two hidden layers are therefore regarded as different models in the following discussion.

Table 7.2 displays the ROC AUC for the Logistic Regression and the machine learning techniques, descending. The scores are based on the test set.

Table 7.2 Overview of Results – ROC AUC

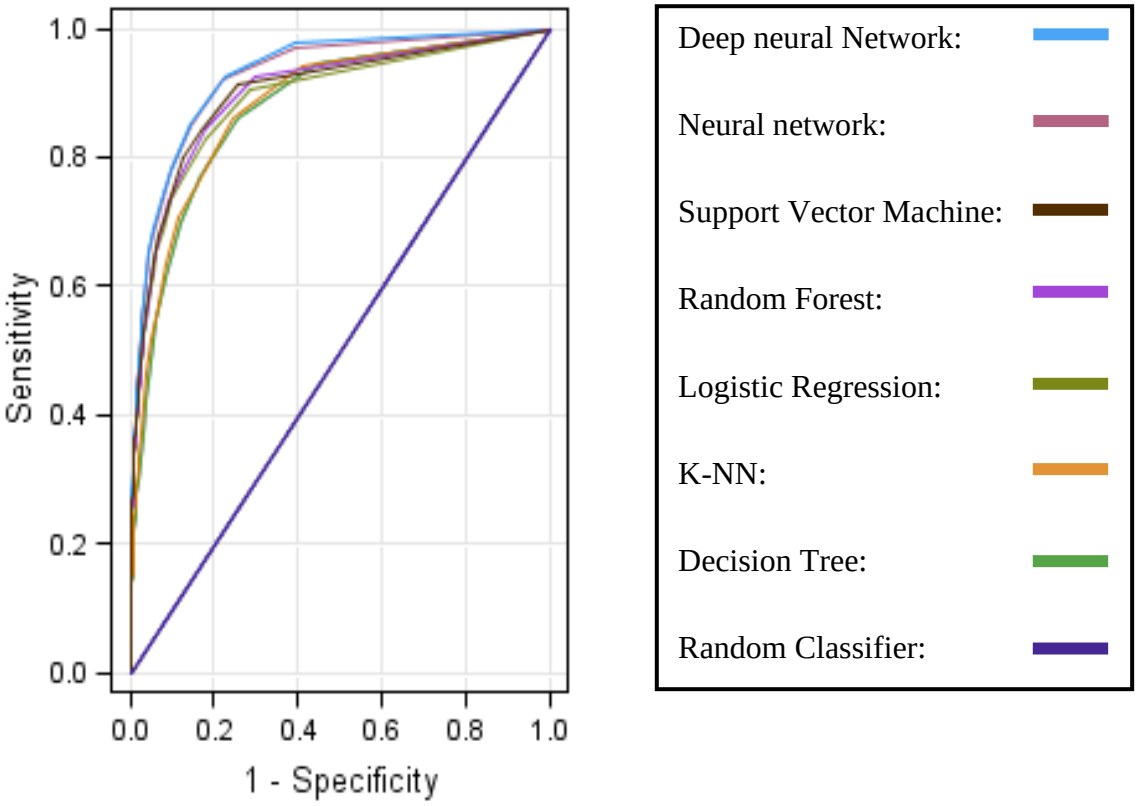
Machine Learning Technique	**AUC ROC**
Deep neural network (two layers)	0,903
Neural network	0,902
Support Vector Machine	0,898
Random Forest	0,897
Logistic regression	0,883
Gradient boosting	0,882
K-NN	0,795
Decision Tree	0,732

As can be seen from the table, the NNs perform best. Interestingly, all machine learning techniques except the Decision Tree and K-NN perform about equally well. The Decision Tree performs much worse, which is expected, since it is a very simple machine learning technique. Indeed, the decision tree is so simple that one cannot blame it for the same black-box properties as the other techniques. Also, one would expect the Random Forest to outperform the Decision Tree, since the former is a more advanced and general version of the latter. As seen from the table, The K-NN algorithm has relatively low predictive power. The NN performs slightly better when it is deep, but anyhow it obtains a high ROC AUC.

The ability of the classifiers to discriminate between customers of different quality is evaluated with the ROC chart. The chart it set up such that the False Positive Rate (FPR) and the True Positive Rate (TPR) is plotted against each other, with different thresholds. The ROC chart for all classifiers is illustrated in Figure 7.1. Since the classifiers perform roughly equally well, with most classifiers achieving a ROC AUC between 0.88 and 0.90, it is difficult to clearly

see the difference. Still, one might observe that the deep neural network (light blue) consistently has a higher TPR (Sensitivity) for a given FPR (1-Specificity) than the other classifiers. Also, it is clear from the figure that the Decision Tree and the K-NN perform worst.

Figure 7.1: ROC Chart for all Classification Techniques



7.3 Confusion Matrix

Table 7.3 displays the confusion matrix for the deep NN, the SVM and the logistic regression. As can be seen, the deep NN has the highest number of true positives and true negatives, which is not surprising given that it is the best classifier. Notice that SAS has only used a sample of the observations to build the confusion matrix (hence the low number of observations).

Table 7.3 Confusion Matrix of Best Classifiers and Logistic Regression

		Actual					
		Deep NN:		**NN:**		**Logistic regression**	
		Default	non-default	Default	non-default	Default	non-default
Prediction	Default	67	55	84	95	64	110
	non-default	673	36210	656	36170	676	36155

7.4 Impact for the Firm

The result of the analysis has been that some of the machine learning techniques, most notably the deep NN, perform slightly better than the logistic regression. The question for the financial institution becomes whether this is of any relevance. As explained, the main drawback of the machine learning models is that they are difficult to interpret. The "Right to explanation" is thus not satisfied, as it would be difficult to explain exactly why the applicant was denied a loan. Under Recital 71 of the GDPR, the data subject "should have the right … to obtain an explanation of the decision reached" (GDPR, 2016). The right to explanation is only one of the challenges with advanced machine learning models, the black-box issue is a more general problem. Some of the machine learning models are not difficult to interpret and may give exact answers to "right to explanation" questions. The simple Decision Tree would go under this category. But as seen in the previous analysis, this model performs much worse than a Logistic Regression, and is thus not relevant. The only models performing better than the Logistic Regression are also very complex.

When applicants in Norwegian banks satisfy the standards for debt to equity ratio and debt to income ratio, they are given a loan unconditional on other variables. In other words, the banks do not bother with predicting individual-level probability of default, exposure at default and loss given default to estimate individual-level expected loss. The banks believe that customers satisfying the criteria on debt to equity and debt to income are probability solvent enough to bother with individual-level estimation of expected loss. From a theoretical standpoint, one would believe that banks do this exercise on each applicant, and only issue loans for customers with expected interest rate income above expected loss.

The field of Explainable AI (XAI) is still young, but in the future we may have come a long way in designing advanced models that are also explainable. Still, some authors believe that there is an inherent tradeoff between interpretability and accuracy, which limits the potential for XAI (Theodorou, Wortham, & Bryson, 2017). Still, If the field of XAI advances any further, it should be feasible to use advanced machine learning techniques in the PD calculation as part of the IRB approach, and perhaps also in individual-level prediction of expected loss.

8. Conclusion

In this master thesis the incremental power of machine learning techniques over Logistic Regression estimation has been analysed. The same WoE variables were used for all classifiers to make results comparable. The ROC AUC for the best classifier was found to be only 0,02 above the Logistic Regression.

The results of the analysis may suggest a ceiling for the predictive performance of the Logistic Regression model. The Logistic Regression model may be used for PD estimation in many years to come, and it is useful to know about how well it might perform. There is no reason to believe that it should perform better than the advanced black-box models such as the deep NN, and the results therefore work well as an estimation of the ceiling. If that argument holds, the model made by WoE transformation and Logistic Regression is very well specified, since it performs almost equally well as the most advanced model in this analysis.

Regardless of the above it is difficult to argue for the introduction of more advanced machine learning models when the difference in ROC AUC above the logistic regression is 0,02. Indeed, if there are any significant non-linear effects in the variables, it seems like the WOE transformation before the logistic regression captures it well.

References

Anderson, R. (2007). *The Credit Scoring Toolkit: Theory and Practice for Retail Credit Risk Management and Decision Automation.* USA: Oxford University Press.

Baesens, B. (2003). Developing Intelligent Systems for Credit Scoring Using Machine Learning Techniques. *Doctoral Thesis no 180 Faculteit Economische en Toegepaste Economische Wetebnschappen, Katholieke Universiteit, Leuven*.

Baesens, B., & Van Gestel, T. (2009). *Credit Risk Management. Basic Concepts: financial risk components, rating analysis, models, economic and regulatory capital.* New York: Oxford University Press Inc.

Basel. (2005). *Basel Committee on Banking Supervision: International Convergence of Capital Measurement and Capital Standards: A Revised Framework.* Bank for International Settlements.

BIS. (2006). *International Convergence of Capital Measurement and Capital Standards.* Basel: Bank for International Settlements.

Boyle, M., Crook, J., Hamilton, R., & Thomas, L. (1992). *Methods for credit scoring applied to slow payers: Credit Scoring and Credit Control.* Oxford: Oxford University Press.

Crook, J. N., Edelman, D. B., & Thomas, L. C. (2007). Recent developments in consumer credit risk assessment. *European Journal of Operational Research*, 1447–1465.

Desai, V., Crook, J., & Overstreet, G. (1997). A comparison of neural networks and linear scoring models in the credit union environment. *IMA Journal of Mathematics Applied in Business and Industry*, 323–346.

Finanstilsynet . (2019). *God praksis ved brug af superviseret machine.* Danmark: Finanstilsynet.

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 1189-1232.

Friedman, J. H. (2002). Stochastic Gradient Boosting. *Computational Statistics and Data Analysis*, 367-378.

GDPR. (2016). *Recitals.* GDPR.

Hand, D. J. (2006). Classifier Technology and the Illusion of Progress. *Statistical Science*, 1-14.

Henley, W. E. (1995). Statistical Aspects of Credit Scoring. *Ph. D Thesis*.

Konishi, S., & Kitagawa, G. (2008). *Information Criteria and Statistical Modeling* . New York: 8 Springer Science+Business Media.

Krauss, A. (2014). *Recent methods from statistics and machine learning for credit scoring.* Göttingen, Germany: Cuvillier Verlag.

Lee, T.-S., Chiu, C.-C., Lu, C.-J., & Chen, I.-F. (2002). Credit scoring using the hybrid neural discriminant technique. *Expert Systems with Applications*, 245–254.

Lendo. (2019, November 15). *Slik får du lån til sekundærbolig*. Retrieved from https://www.lendo.no/boliglan/lan-til-sekundaerbolig/

Mueller, J. P., & Massaron, L. (2016). *Machine Learning For Dummies.* United States: John Wiley & Sons.

Ong, C.-S., Huang, J.-J., & Tzeng, G.-H. (2005). Building credit scoring systems using genetic programming. *Expert Systems with Applications*, 41–47.

Provost, F., & Fawcett, T. (2015). *Data Science for Business.* United States: O'Reilly Media.

SAS. (2017). *SAS® Enterprise Miner™ 14.3: Reference Help.* Cary, NC: SAS Institute Inc. Retrieved from https://documentation.sas.com/?docsetId=emref&docsetTarget=titlepage.htm&docsetVersion=14.3&locale=en.

SAS Institute Inc. (2013, December 17). *Base SAS® 9.4 Procedures Guide: High-Performance Procedures.* Cary, NC: SAS Institute Inc. Retrieved from support.sas.com: http://support.sas.com/documentation/cdl/en/prochp/66704/HTML/default/viewer.htm#prochp_hpbin_details02.htm

Siddiqi, N. (2006). *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring.* Hoboken, New Jersey: John Wiley & Sons.

SSB. (2019, September 10). *Konsumprisindeksen*. Retrieved from SSB: https://www.ssb.no/kpi

SSB. (2019, September 18). *Nasjonalregnskap*. Retrieved from SSB: https://www.ssb.no/statbank/table/09786

Theodorou, A., Wortham, R. H., & Bryson, J. J. (2017). Designing and implementing transparency for real time inspection of autonomous robots. *Connection Science*, 230-241.

Thomas, L. C. (2000). A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. *International Journal of Forecasting*, 149–172.

West, D. (2000). Neural network credit scoring models. *Computers and Operational Research*, 1131–1152.

Yobas, M. B., Crook, J. N., & Ross, P. (2000). Credit scoring using neural and evolutionary techniques. *IMA Journal of Mathematics Applied in Business and Industry*, 111–125.

Appendix

Appendix 1 – Variable descriptions left out of chapter 4.9

Employment Variable

There is one employment variable – "Employment". This is a nominal variable and can take any of the following values: NW, LO, ST, YA, WI. There is a lack of information conserning what these abbreviations stand for, which of course is a cause for consern. However, because the variable might be of significant importance, it is included as a variable to be tested.

Relationship Status Variable

There is one relationship status variable. This nominal variable can take on the values:

- EN: Engaged
- WI: Widow/widower
- SE: Separation
- DI: Divorce
- CP: Common-law partner
- UM: Unmarried
- MA: Married
- Missing

Income Variables

There are four income variables in the dataset. The definition of each income variable is given below.

Net_income: The sum of the debtor's and co-debtor's net income. The net income for the last three years is found from the tax returns, which are official. The most recent year's income is used.

Net_income_max: Same as above, but the maximum net income for the last three years is used.

Net_income_co: Co-debtor's net income. The net income for the last three years is found from the tax returns. The most recent year's income is used.

Net_income_main: Same as above, but for the main debtor.

Net Worth Variables

There are four net worth variables. The definition of each variable is given below:

Net_worth: This variable takes the sum of the main debtor's and co-debtor's net worth. The tree previous years are examined, and the nearest year with values for both main debtor and co-debtor is used.

Net_worth_AVG: Same as above, but the average value.

Net_worth_co: This is the co-debtor's net worth in the tax returns for one of the three previous years (the nearest year without missing value is chosen).

Net_worth_main: Same as above, but for the main debtor.

Appendix 2 – Choosing only the 12-month variables

	3	6	9	12
Savings_balance_MIN	0.815	0.787	0.736	0.732
Savings_balance_MED	0.813	0.769	0.76	0.759
Savings_balance_SUM	0.768	0.765	0.733	0.75
Savings_balance_AVG	0.765	0.738	0.733	0.737
Savings_balance_MAX	0.724	0.698	0.646	0.625
Savings_balance_STD	0.406	0.501	0.491	0.511

	3	6	9	12
Fixed_in_trans_MIN	0.254	0.316	0.345	0.352
Fixed_in_trans_MED	0.172	0.148	0.157	0.15
Fixed_in_trans_SUM	0.168	0.141	0.149	0.145
Fixed_in_trans_AVG	0.161	0.145	0.149	0.132
Fixed_in_trans_MAX	0.121	0.103	0.103	0.112
Fixed_in_trans_STD	Below	Below	0.105	0.766

	3	6	9	12
Variable_in_trans_MIN	Below	Below	Below	Below
Variable_in_trans_MED	Below	Below	Below	Below
Variable_in_trans_SUM	Below	0.11	0.13	0.158
Variable_in_trans_AVG	Below	0.111	0.133	0.154
Variable_in_trans_MAX	Below	0.1	0.1	0.114
Variable_in_trans_STD	Below	0.101	0.102	0.12

	3	6	9	12
Fixed_out_trans_MIN	Below	Below	Below	0.104
Fixed_out_trans_MED	0.122	0.105	Below	Below
Fixed_out_trans_SUM	0.11	Below	0.1	0.105
Fixed_out_trans_AVG	0.11	Below	0.102	0.108
Fixed_out_trans_MAX	0.169	0.212	0.251	0.25
Fixed_out_trans_STD	Below	0.1	0.102	0.108

	3	6	9	12
Varibale_out_trans_MIN	0.161	0.183	0.18	0.188
Varibale_out_trans_MED	0.151	0.14	0.134	0.143
Varibale_out_trans_SUM	0.154	0.15	0.133	0.146
Varibale_out_trans_AVG	0.151	0.159	0.15	0.15
Varibale_out_trans_MAX	0.153	0.147	0.143	0.148
Varibale_out_trans_STD	0.153	0.197	0.237	0.239

Appendix 3 – Full list of hyperparameters

NN:	
Use inverse priors:	No
Create validation	No
Input standardization	Range
Architecture	Two layers
Number of hidden neurons	20
Target standardization	Range
Target activation function:	Identity
Target error function	Normal
Number of tries	2
Maximum iterations	300
Use missing as level	No
Maximum number of links	1000
RF:	
Maximum number of trees:	100
Seed	12345
Type of sample	Proportion
Proportion of obs in each sample	0,6
Maximum depth	30
Missing values	Use in search
Maximum use in search	1
Significance level	0,05
Maximum categories in split search	30
Minimum category size	5
Exhaustive	5000
Method for leaf size	Default
Variable selection	Yes
Variable importance method	Loss reduction
Gradient Boosting:	
N iterations	200
Seed	12345
Shrinkage	0,1
Train proportion	60
Huber M-regression	No
Maximum branch	30
Maximum depth	40
Minimum categorical size	5
Reuse variable	1
Categorical bins	30
Interval bins	100
Missing values	Use in search
Performance	Disk
Leaf fraction	0,1
Number of surrogate rules	0
Exhaustive	5000
Node sample	20 000
Assessment measure	Decision
Subseries	Best assessment value
Create H statistic	No
Variable selection	Yes
Observation Based Importance	No

SVM	
Maximum iterations	80
Use missing as level	No
Tolerance	1.0E-6
Penalty	1
Optimization method	Interior Point
KNN	
Method	RD-Tree
Number of neighbors	16
Epsilon	0
Number of buckets	8
Weighted	Yes
Create nodes	No
Create neighbor variables	Yes
Decision tree	
Interval target criterion	ProbF
Nominal target criterion	Entropy
Ordinal target criterion	Entropy
Significance level	0,2
Missing values	Use in search
Use input once	No
Maximum branch	10
Maximum depth	50
Minimum categorical size	5
Leaf size	5
Number of rules	5
Number of surrugate rules	5
Use decisions	No
Use priors	No
Exhaustive	5000
Node sample	20 000
Method	Assessment
Assessment measure	Decision
Observation based importance	No
Bonferroni adjustment	Yes
Time of Bonferroni adjustment	Before
Inputs	No
Depth adjustment	Yes
Leaf variable	Yes
Create sample	Default
Performance	Disk
Variable selection	Yes
Leaf Role	Segment

Scorecard (log. Reg)	
Analysis variables	WOE
Freeze scorecard points	None
Output variables	Complete
Intercept based scorecard	No
Reverse scorecard	No
Odds	50
Scorecard points	200
Points to double odds	20
Scorecard type	Detailed
Precision	0
Bucketing method	Min/max distribution
Number of buckets	25
Use indeterminate values	No
Revenue accepted good	1000
Cost accepted bad	50000
Current approval rate	70
Current event rate	2,5
Generate characteristic analysis	No
Selection model	Stepwise
Criterion	AIC